US005442784A

United States Patent [19]

Powers et al.

[11] **Patent Number: 5,442,784**

[45] **Date of Patent: * Aug. 15, 1995**

[54] **DATA MANAGEMENT SYSTEM FOR BUILDING A DATABASE WITH MULTI-DIMENSIONAL SEARCH TREE NODES**

[75] Inventors: **Frederick A. Powers**, Sudbury; **Stanley R. Zanarotti**, Cambridge, both of Mass.

[73] Assignee: **Dimensional Insight, Inc.**, Burlington, Mass.

[*] Notice: The portion of the term of this patent subsequent to Oct. 26, 2010 has been disclaimed.

[21] Appl. No.: **78,396**

[22] Filed: **Jun. 17, 1993**

Related U.S. Application Data

[62] Division of Ser. No. 495,360, Mar. 16, 1990, Pat. No. 5,257,365.

[51] **Int. Cl.**[6] **G06F 17/30**

[52] **U.S. Cl.** **395/600;** 364/251.6; 364/252.3; 364/282.1; 364/283.2; 364/DIG. 1

[58] **Field of Search** 395/600

[56] **References Cited**

U.S. PATENT DOCUMENTS

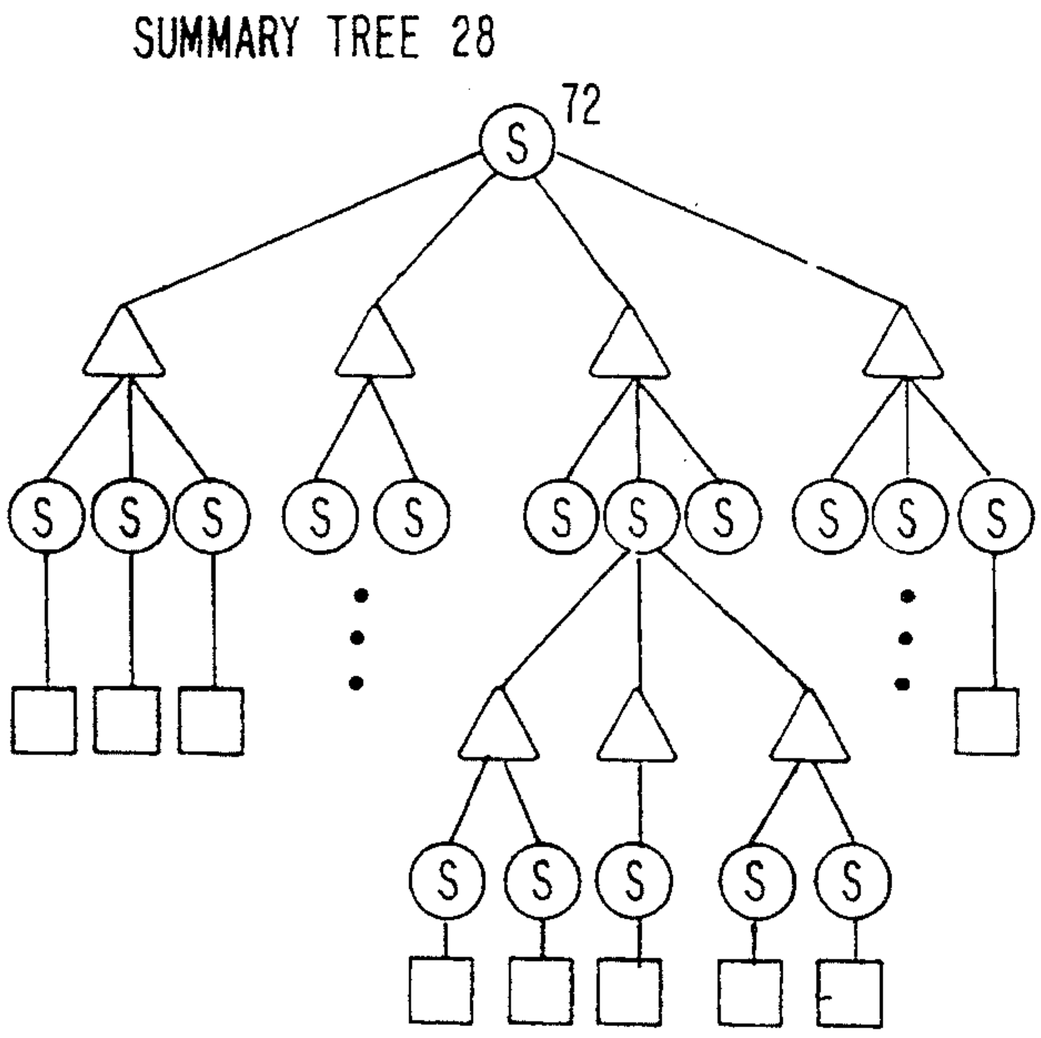
3,670,310	6/1972	Bharwani et al.	395/600
4,318,184	3/1982	Millett et al.	395/600
4,468,732	8/1984	Raver	395/600
4,606,002	8/1986	Waisman et al.	395/600
4,611,298	9/1986	Schuldt	395/600
4,631,664	12/1986	Bachman	395/600
4,817,036	3/1989	Millett et al.	395/600
4,817,050	3/1989	Komatsu et al.	395/600
4,829,427	5/1989	Green	395/600
4,939,689	7/1990	Davis et al.	395/600
4,961,139	10/1990	Hong et al.	395/600
5,201,047	4/1993	Maki et al.	395/600
5,201,048	4/1993	Coulter et al.	395/600
5,237,678	8/1993	Kuechler et al.	395/600

Primary Examiner—Thomas G. Black
Assistant Examiner—Paul Harrity
Attorney, Agent, or Firm—Gary D. Clapp

[57] **ABSTRACT**

The subject invention is directed to a database system for organizing large amounts of data to be accessed by a digital computer. More particularly, a free form type database, in the form of a summarized, multikey tree, is built from files stored on the computer. After a building operation, the user obtains specified information by using the summarized database. Information in the files is divided into three categories; that is, a dimension field which comprises data to be organized, a summary field which comprises a numeric quantity on which calculations can be performed, and a non-summary field which comprises other information associated with an input record. The internal nodes of the tree summarize and organize sets of input records. Methods are provided for reducing the amount of storage space used by cutting off the tree when the size of sets go below a given threshold, and sharing parts of the tree so that each record does not appear n! times in the database.

7 Claims, 9 Drawing Sheets



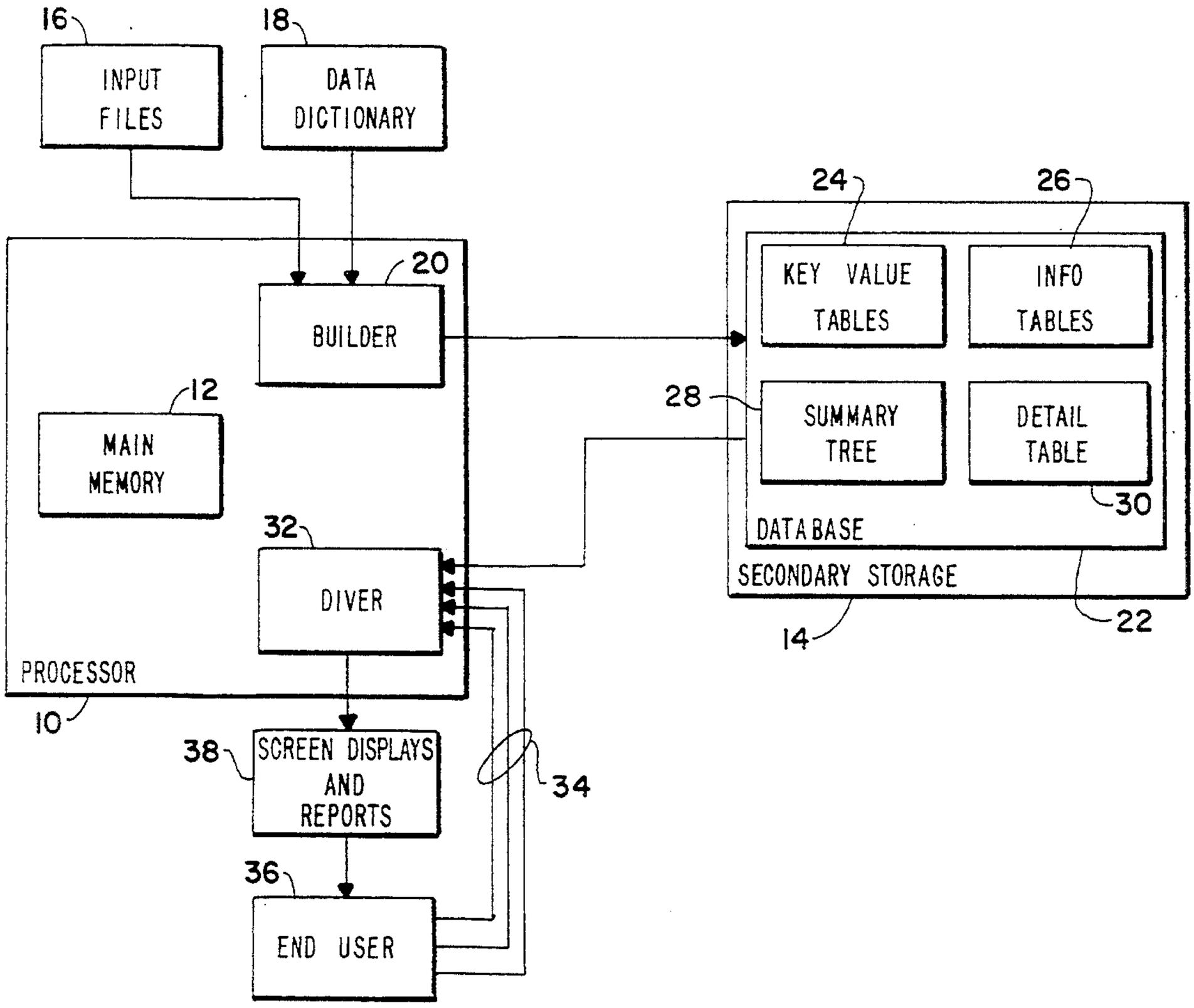
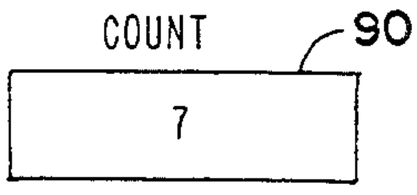


Fig. 1



SUMMARY FIELD	SUM (96)	SUM OF SQUARES (98)	MINIMUM (100)	MAXIMUM (102)
92 SALARY	231500	9744710000	18700	72800
94 AGE	199	6167	21	48

Fig. 6

40
42
NAME FIELD TYPE ASSOCIATED DIMENSION
44

EMPLOYEE FILE ← 50

FIELD 46

EMPLOYEE ID	NAME	SEX	ZIPCODE	ADDRESS	SALARY	AGE	DEPT ID
DIMENSION	NON-SUMMARY	DEMENSION	DIMENSION	NON-SUMMARY	SUMMARY	SUMMARY	DIMENSION
--	DETAIL	--	--	DETAIL	--	--	--
204	SMITH,JOHN	M	01776	11 ELM ST.	72800	48	ADMIN
208	JONES,DAVE	M	01890	77 MAIN ST.	25600	26	ADMIN
225	TAYLOR,JANE	F	01779	18 TREMONT	32500	30	FIN
210	LEE,SCOTT	M	02046	131 BEACON	18700	21	SHIP
320	FUCHS,MARY	F	01890	2013 MAPLE	31800	27	R&D
234	BOWLER,STAN	M	02046	170 CAMBRIDGE	28800	26	R&D
221	DALTON,ANDRE	M	02139	300 VASSAR	23300	21	R&D
•	•	•	•	•	•	•	•

DATA DICTIONARY ↳ 18

RECORD <---- 48

Figure 2B

40
42
NAME FIELD TYPE ASSOCIATED DIMENSION
44

DEPARTMENT FILE ← 52

DEPT ID	DEPT NAME	MANAGER
DIMENSION	NON-SUMMARY	NON-SUMMARY
	DEPT ID	DEPT ID
ADMIN	ADMINSTRATION	SMITH,JOHN
FIN	FINANCE	TAYLOR,JANE
R&D	RES.&DEVL.	WOOD,JOSEPH
SHIP	SHIPPING	FOX,SEAN

Figure 2A

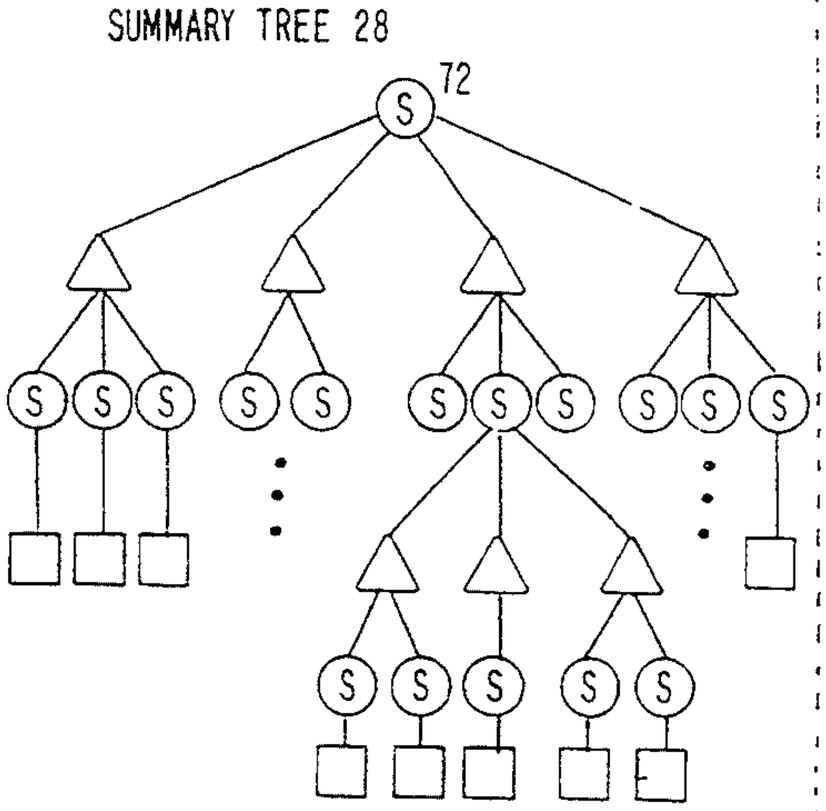


Figure 3A

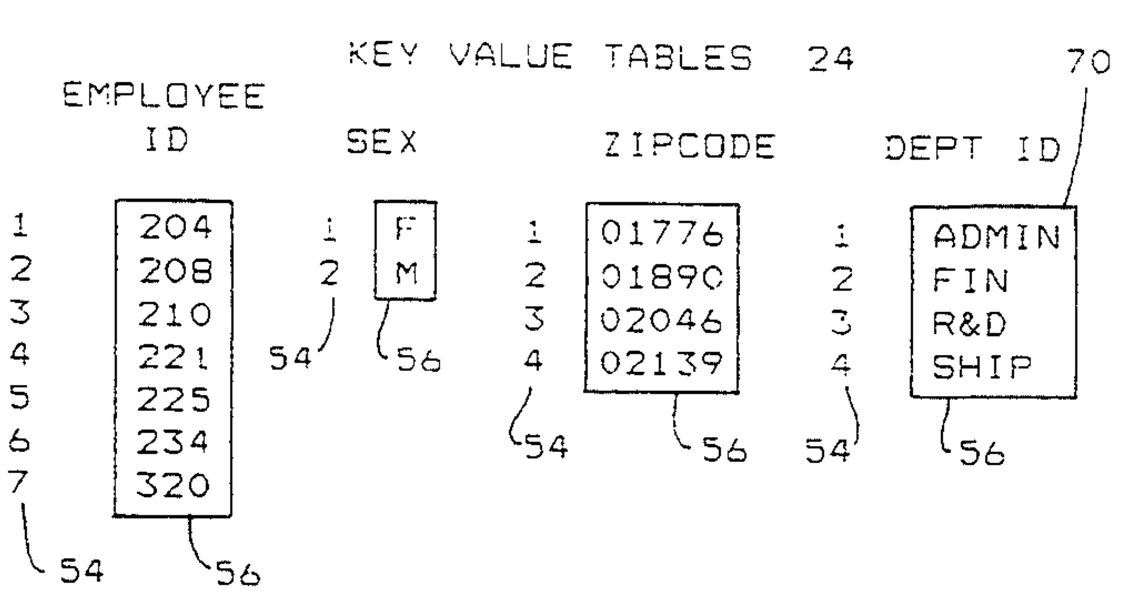


Figure 3B

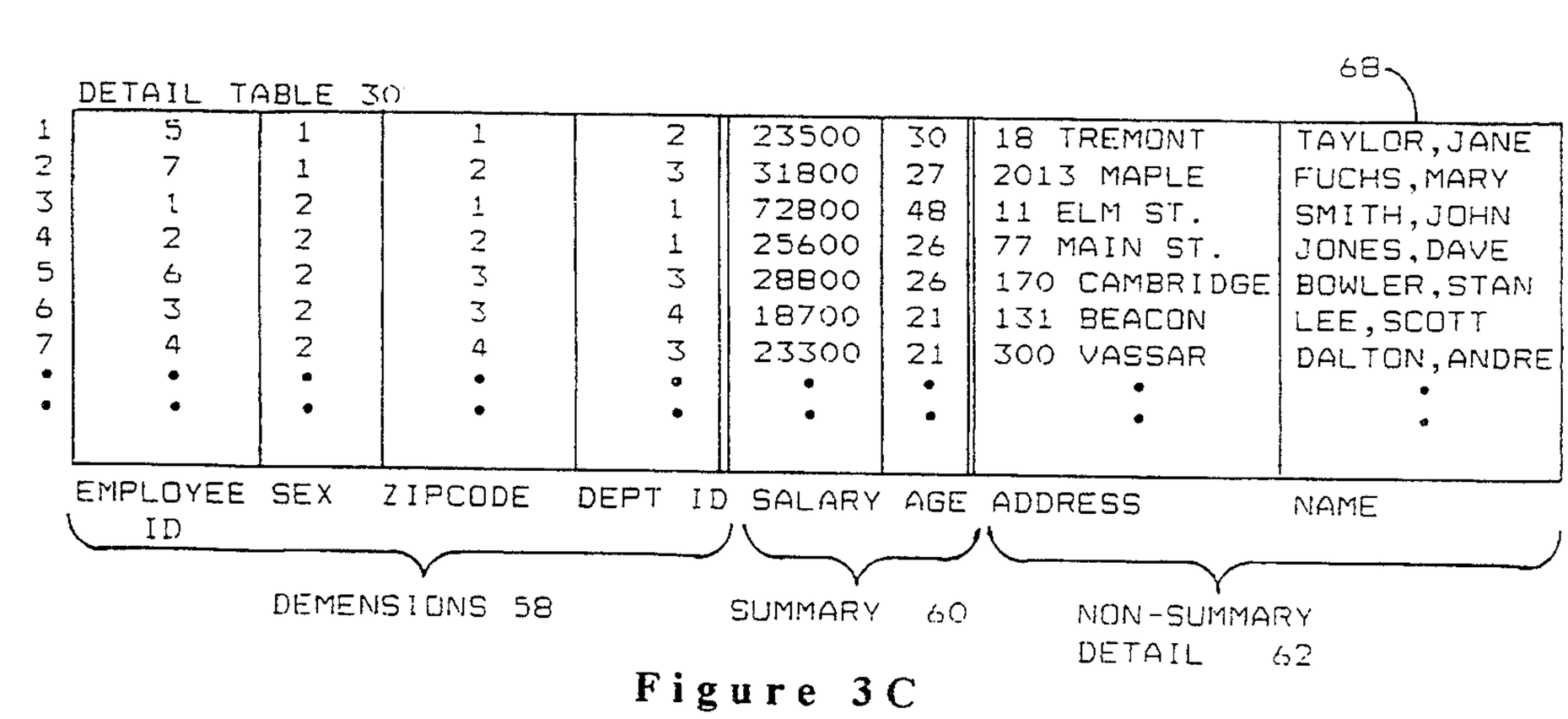


Figure 3C

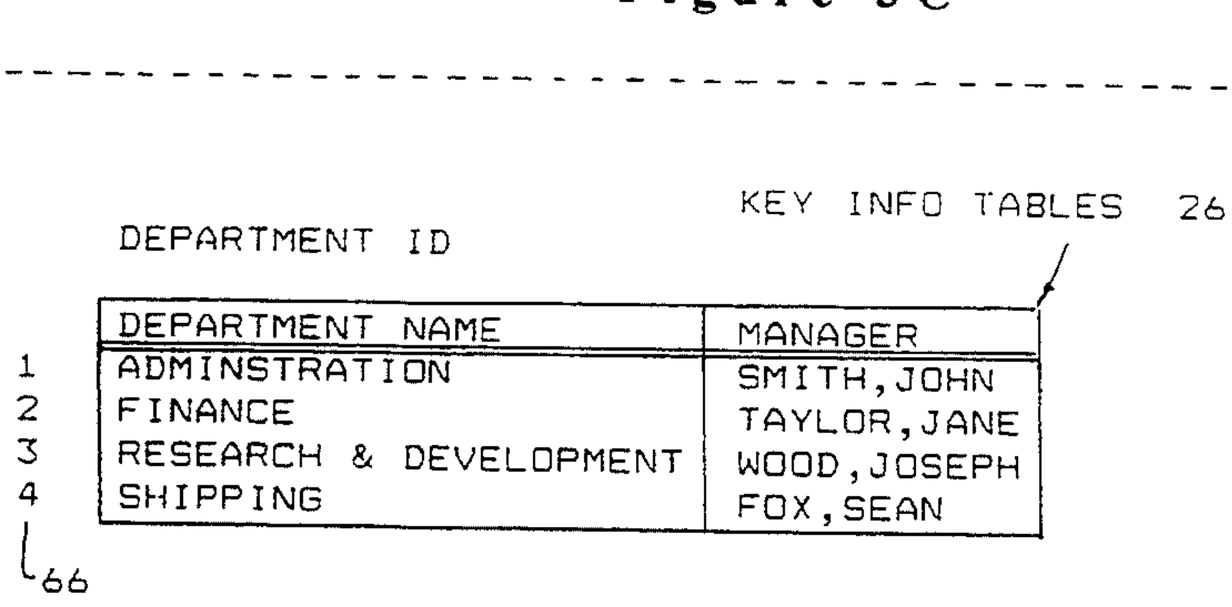


Figure 3D

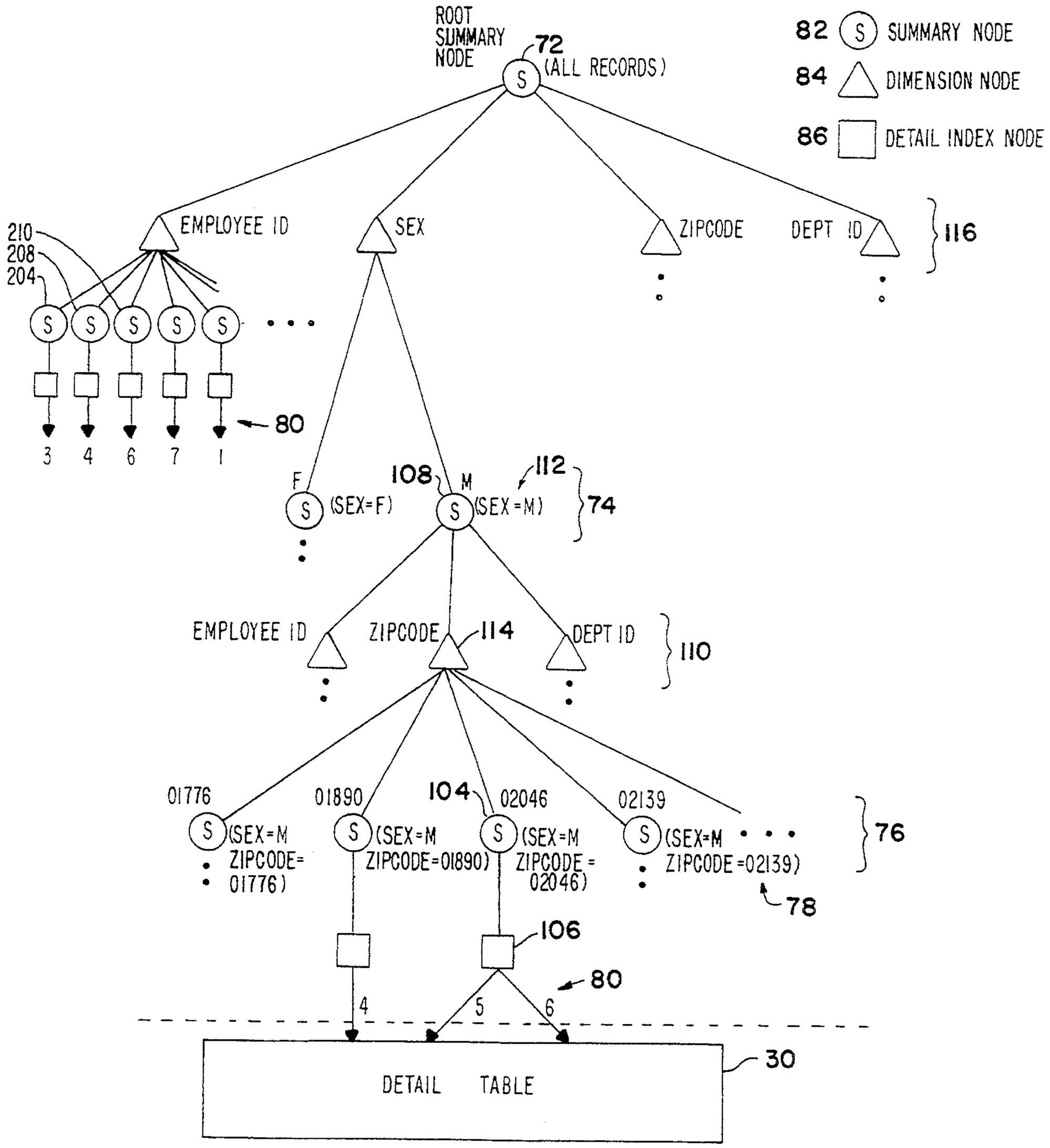

ROOT SUMMARY NODE
72
(ALL RECORDS)
82 S SUMMARY NODE
84 DIMENSION NODE
86 DETAIL INDEX NODE
EMPLOYEE ID
SEX
ZIPCODE
DEPT ID
116
210
208
204
80
3 4 6 7 1
F
(SEX=F)
108
M
112
(SEX=M)
74
EMPLOYEE ID
ZIPCODE
114
DEPT ID
110
01776
01890
104
02046
02139
(SEX=M ZIPCODE= 01776)
(SEX=M ZIPCODE=01890)
(SEX=M ZIPCODE= 02046)
(SEX=M ZIPCODE=02139)
76
78
106
80
4
5
6
30
DETAIL TABLE

Fig. 4

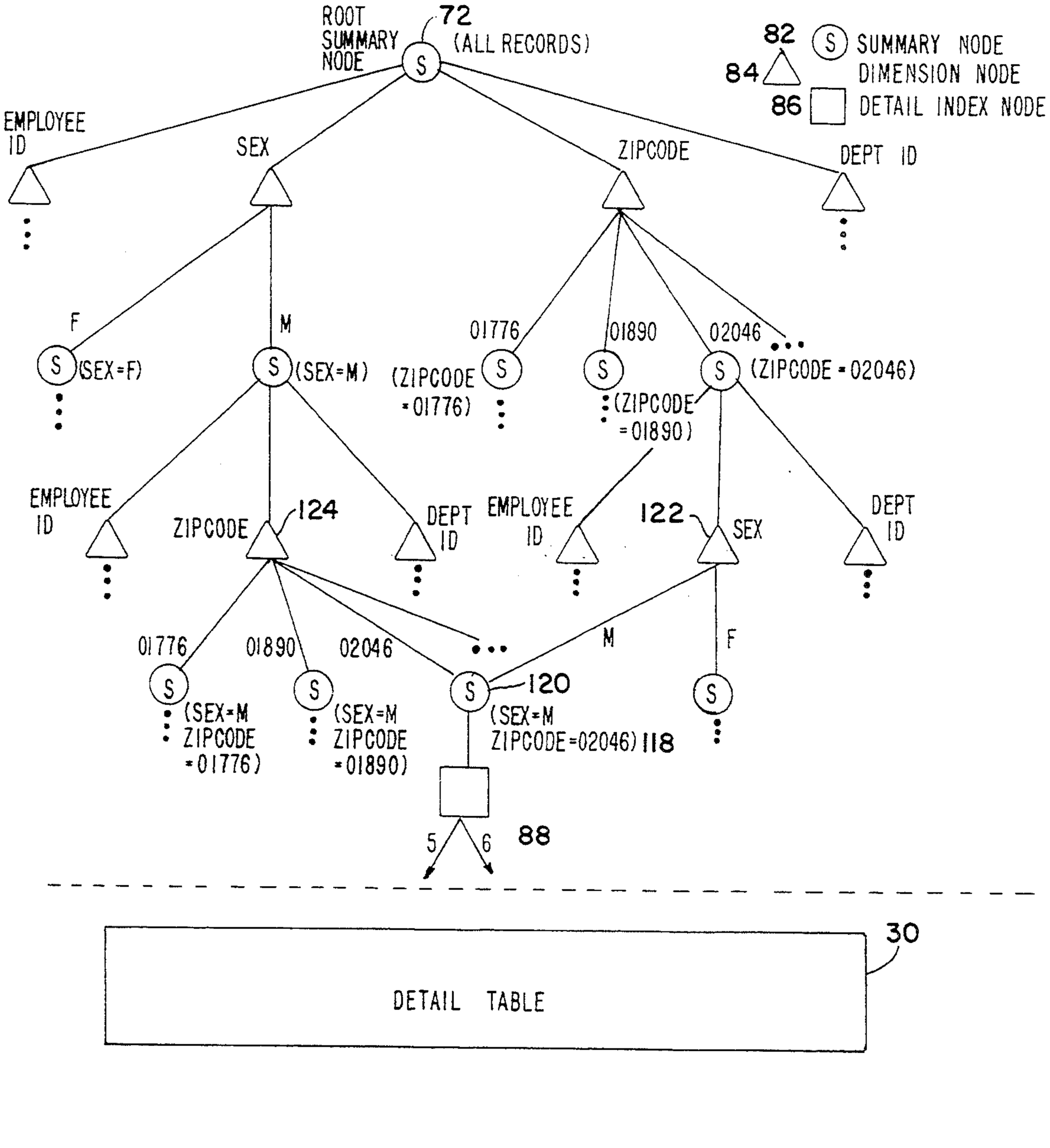

ROOT SUMMARY NODE
72
(ALL RECORDS)
82 S SUMMARY NODE
84 DIMENSION NODE
86 DETAIL INDEX NODE
EMPLOYEE ID
SEX
ZIPCODE
DEPT ID
F
M
01776
01890
02046
(SEX=F)
(SEX=M)
(ZIPCODE =01776)
(ZIPCODE =01890)
(ZIPCODE=02046)
EMPLOYEE ID
ZIPCODE
124
DEPT ID
EMPLOYEE ID
122
SEX
DEPT ID
01776
01890
02046
M
F
120
(SEX=M ZIPCODE =01776)
(SEX=M ZIPCODE =01890)
(SEX=M ZIPCODE=02046)
118
5
6
88
DETAIL TABLE
30

Fig. 5

ROOT SUMMARY NODE 72

COUNT 90	SALARY 92 SUM	SUM OF SQUARES	MIN	MAX	AGE 94 SUM	SUM OF SQUARES	MIN	MAX
7	231506	7744710000	18700	72800	199	6167	21	48

Figure 7A

DIMENSION POINTERS 126	EMPLOYEE ID	SEX	ZIPCODE	DEPT ID

EMPLOYEE ID	SALARY SUM OF SQUARES	MIN	MAX	AGE SUM	SUM OF SQUARES	MIN	MAX
48	6356090000	32500	72800	78	3209	30	48
27	1666600000	25500	31800	53	1405	26	27
26	1179100000	18700	28800	47	1117	21	26
...	...	...	...	...	...	...	...

138 (ZIPCODE) DIMENSION VALUE — 90 COUNT — SUM

128			
130 → 1	2	105300	63
132 → 2	2	57400	16
134 → 3	2	47500	11
...	...	...	

136

Figure 7B

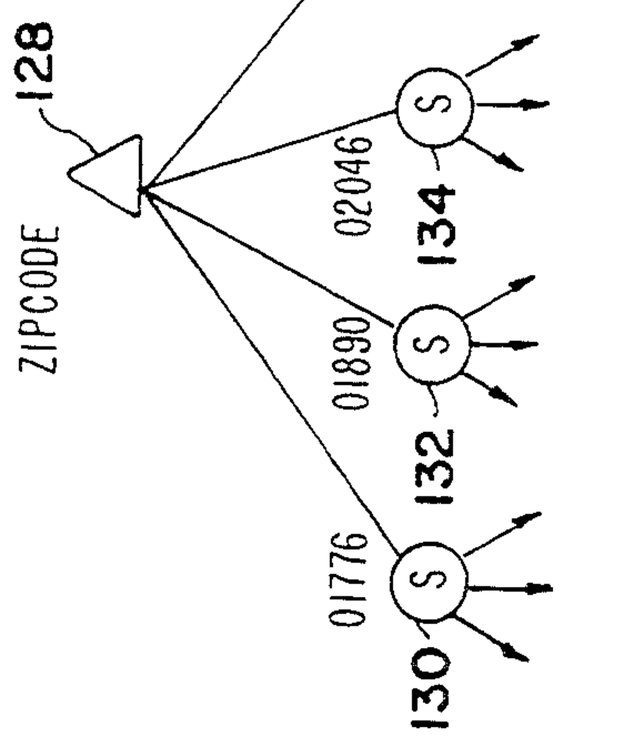


Figure 7C

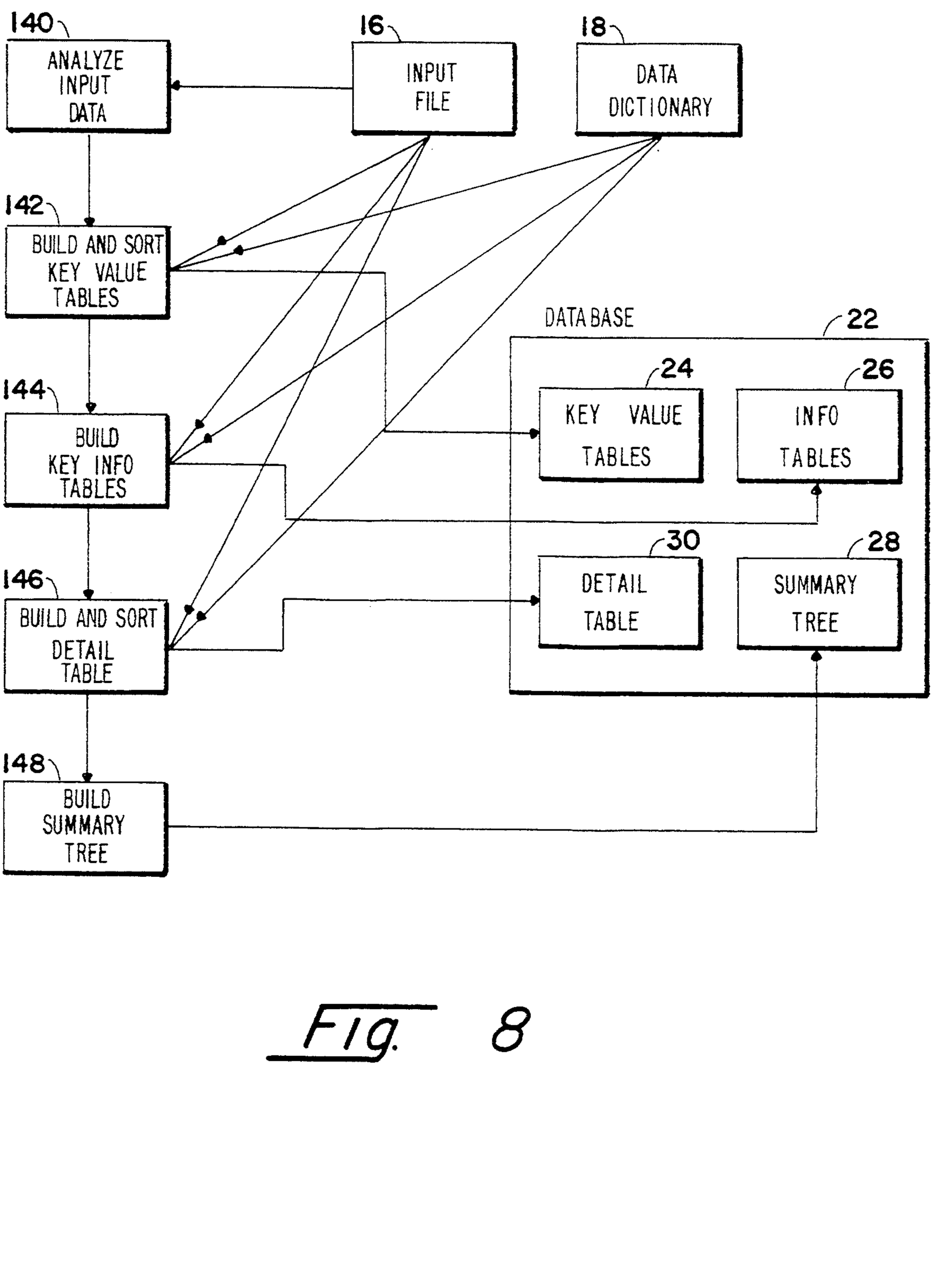

140
ANALYZE INPUT DATA
16
INPUT FILE
18
DATA DICTIONARY
142
BUILD AND SORT KEY VALUE TABLES
144
BUILD KEY INFO TABLES
146
BUILD AND SORT DETAIL TABLE
148
BUILD SUMMARY TREE
DATA BASE
22
24
KEY VALUE TABLES
26
INFO TABLES
30
DETAIL TABLE
28
SUMMARY TREE

Fig. 8

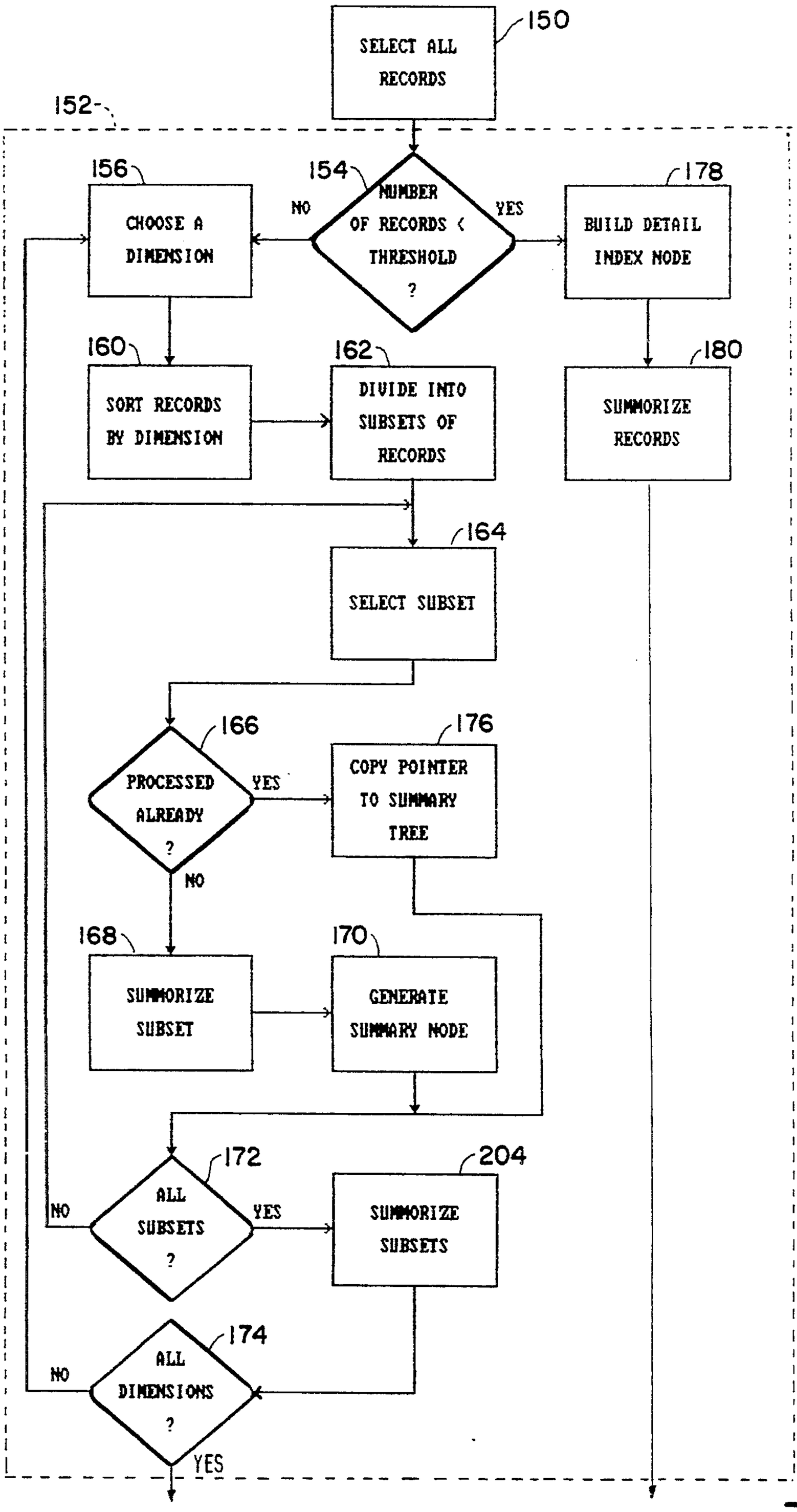

SELECT ALL RECORDS
150
152
156
CHOOSE A DIMENSION
154
NUMBER OF RECORDS < THRESHOLD ?
NO
YES
178
BUILD DETAIL INDEX NODE
160
SORT RECORDS BY DIMENSION
162
DIVIDE INTO SUBSETS OF RECORDS
180
SUMMORIZE RECORDS
164
SELECT SUBSET
166
PROCESSED ALREADY ?
YES
176
COPY POINTER TO SUMMARY TREE
NO
168
SUMMORIZE SUBSET
170
GENERATE SUMMARY NODE
172
ALL SUBSETS ?
NO
YES
204
SUMMORIZE SUBSETS
174
ALL DIMENSIONS ?
NO
YES

Fig. 9

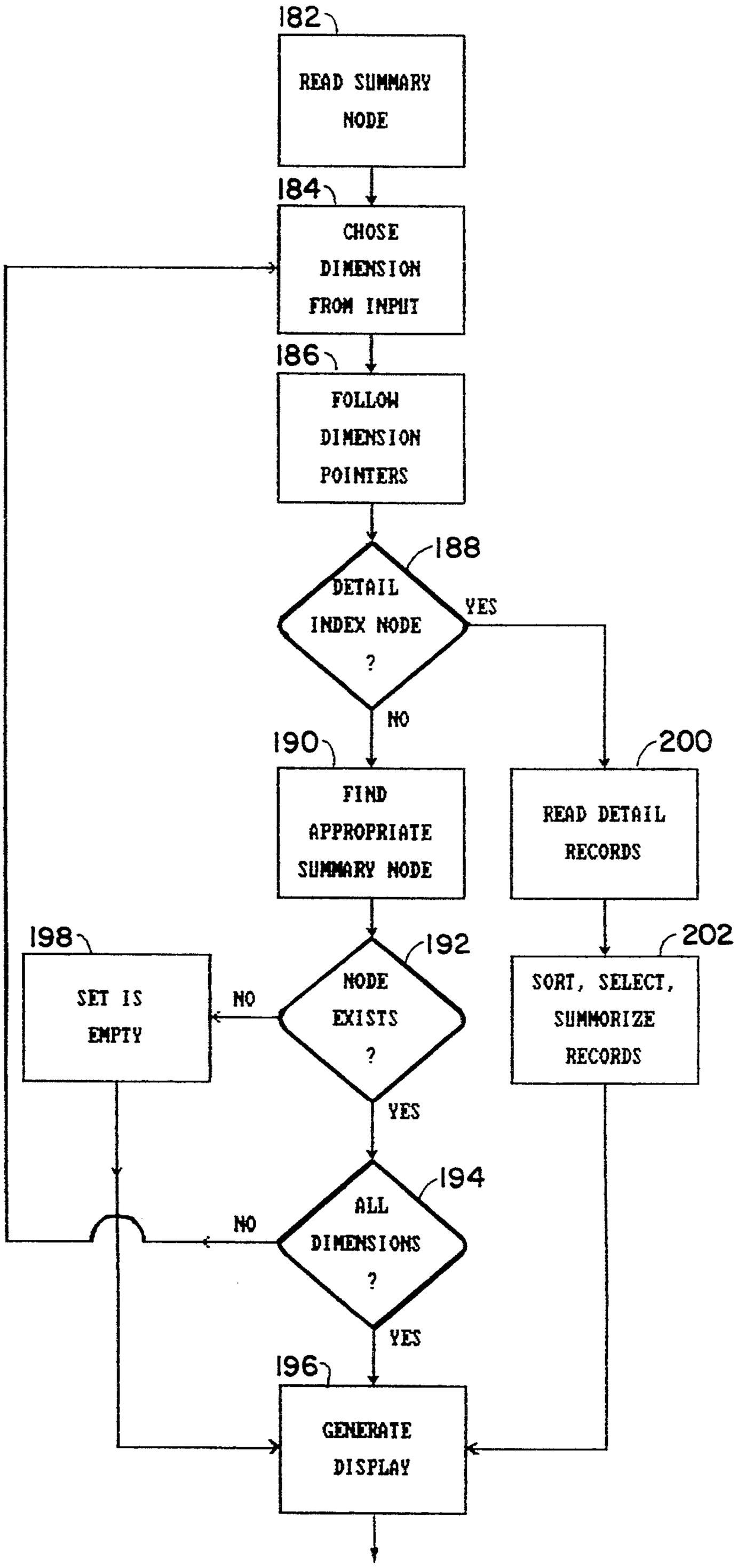

182
READ SUMMARY NODE
184
CHOSE DIMENSION FROM INPUT
186
FOLLOW DIMENSION POINTERS
188
DETAIL INDEX NODE ?
YES
NO
190
FIND APPROPRIATE SUMMARY NODE
200
READ DETAIL RECORDS
198
SET IS EMPTY
192
NODE EXISTS ?
NO
YES
202
SORT, SELECT, SUMMORIZE RECORDS
194
ALL DIMENSIONS ?
NO
YES
196
GENERATE DISPLAY

Fig. 10

DATA MANAGEMENT SYSTEM FOR BUILDING A DATABASE WITH MULTI-DIMENSIONAL SEARCH TREE NODES

CROSS REFERENCES TO RELATED APPLICATIONS

The present Patent Application is a Divisional Patent Application from U.S. patent application Ser. No. 07/495,360 for MULTI-DIMENSIONAL SUMMARY DATABASE SYSTEM AND METHOD by Frederick A Powers and Stanley R. Zanarotti, filed Mar. 16, 1990 and since allowed as U.S. Pat. No. 5,257,365 with an issue date of Oct. 26, 1993.

U.S. patent application Ser. No. 08/079,248, still pending, by Frederick A. Powers and Stanley R. Zanarotti for MULTI-DIMENSIONAL SUMMARY DATABASE SYSTEM AND METHOD by Frederick A. Powers and Stanley R. Zanarotti, filed on the same date as the present Patent Application; and,

U.S. patent application Ser. No. 08/079,249, still pending, by Frederick A. Powers and Stanley R. Zanarotti for MULTI-DIMENSIONAL SUMMARY DATABASE SYSTEM AND METHOD by Frederick A. Powers and Stanley R. Zanarotti, filed on the same date as the present Patent Application.

The above referenced U.S. Patent Applications are assigned to the assignee of the present U.S. Patent Application.

REFERENCE PUBLICATIONS "Multi-attribute Retrieval with Combined Indexes", November, 1970, Communications of the ACM, pp 660–665, Vol 13, Number 11

BACKGROUND OF THE INVENTION

1. Field of the invention

This invention relates to methods of storing and accessing data on digital computers, and in particular, to an improved data base system for organizing large amounts of data for fast retrieval and processing.

2. Description of the Prior Art

Databases are used to store large amounts of data in digital computers. To analyze this data, users need to be able to identify sets of records based on a combination of attributes and generate summary information, such as sums, averages, and other statistical functions, for these sets.

Traditional databases may provide support for identifying some of these sets, but not all of them in an efficient manner. Multidimensional databases can provide fast access to more sets, for a small number of attributes. Even so, providing summary information on a set requires accessing all elements of that set, and is a time-consuming operation for large sets thereby delaying interactive queries for this information.

The following U.S. Patents disclose typical database management systems.

U.S. Pat. No. 4,554,631, entitled "Keyword Search Automatic Limiting Method."

U.S. Pat. No. 4,606,002, entitled "B-Tree Structured Data Base Using Spare Array Bit Maps to Store Inverted Lists."

U.S. Pat. No. 4,611,272, entitled "Key-Accessed File Organization."

U.S. Pat. No. 4,468,728, entitled "Data Structure and Search Method for a Data Base Management System."

OBJECTS OF THE INVENTION

Accordingly, it is a primary object of the invention to have an improved database management system.

It is an object of the invention to have a database management system for providing rapid summary information for large sets of records.

SUMMARY OF THE INVENTION

The above objects and advantages are achieved in a preferred embodiment of the present invention. According to the preferred embodiment, a data base management system for storing and accessing data provides fast interactive access to summary information for different sets of input records, where the sets are defined by specifying values for multiple attributes. The method involves calculating a large portion of this summary information and building it into a data structure. At access time, users specify sets by giving values for multiple attributes, and the data structure is searched for the appropriate summary information for that set. If found, the summary information is displayed; otherwise, the summary information is calculated from the records themselves.

The data structure consists of the original data in a relational detail table, a summary tree structure for organizing and summarizing the data along several dimension fields, and key value tables for encoding dimension field values as integers.

Key info tables store information associated with dimension field values, for convenient reference. The summary tree is a search tree where internal nodes describe and summarize numeric fields in sets of records. Nodes deeper in the tree describe more specific sets of records, until the size of the set is smaller than a given threshold, at which point the individual records of the set are indexed by a detail index node. The tree can be represented as a numbered set of tables.

At access time, the user specifies a set of dimension values. The summary tree is walked using these values to find the appropriate node. If the node is a summary node, the summary information is displayed. If the node is a detail index node, the set of records is read from the detail table, and the summary information is calculated from this set.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features which are believed to be characteristic of the invention both as to its organization and method of operation, together with further objects and advantages will be better understood from the following description when considered in connection with the accompanying drawings. It is expressly understood, however, that each of the drawings is given for the purpose of illustration and description only and is not intended as a definition of the limits of the present invention.

FIG. **1**. shows in block diagram form the systems architecture for the invention, illustrating the general process of building and accessing the database.

FIGS. **2A** and **2B**. show sample data files, suitable as input to this invention, and the corresponding data dictionary.

FIGS. **3A**, **3B**, **3C**, and **3D**. give a high-level view of the database, showing the major components.

FIGS. **4**. and **5** show the structure of the summary tree, which is used to access and organize the data along

arbitrary dimension fields. FIG. **5** shows the summary tree with sharing subtrees.

FIG. **6**. shows the contents of a summary node.

FIGS. 7A, 7B, and 7C. show how the summary tree can be represented in a tabular format.

FIG. **8**. shows the builder program, which builds the database.

FIG. **9**. shows how the summary tree portion of the database is built.

FIG. **10**. shows the diver program, and how it accesses the database to provide summary information to the user.

DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention is a method of storing data in the memory of a data processing system and accessing the data in a manner that provides fast interactive access to summary information for different sets of input records, where the sets are defined by specifying values for multiple search keys. The method involves calculating a large portion of this summary information and building it into a database. When this database is accessed, this summary information is available without having to calculate it. Although the process of building the database may be a time-consuming operation, it can be done when the system is off-line. At access time, when an operator is waiting for an answer, a database query can be answered quickly.

FIG. **1**. shows the system architecture of the present invention. The invention generally comprises a processor **10**, a main memory **12**, and secondary storage **14**. The processor **10** is used to run builder **20** and diver **32** programs, using main memory **12** for temporary storage and intermediate results, and using secondary storage **14** for permanent storage. Input data files **16**, described by a data dictionary **18** is fed into the builder program **20**. The builder program **20** produces a database **22**, which is comprised of key value tables **24**, key info tables **26**, a summary tree **28**, and a detail table **30**. The diver program **32**, given dimension selections **34** from an end user **36**, uses the database **22** to produce screen displays and reports **38** to the end user **36**.

FIG. **2**. shows sample input data that can be used as input to the builder program **20**. The input to the builder **20** is modelled as a set of flat files. Each flat file **50**, **52** consists of a number of records **48**, each of which describes a single entity. The records **48** are divided up into different fields **46**, which represent different attributes of the entity. Multiple files are related by using a common field to perform a standard relational database join operation to produce a single logical file, with one record for each entity.

The data dictionary **18** of FIG. **1** is used to describe the fields in the input files, and assigns a type to each of the fields. The data dictionary contains three pieces of information about each field: a field name **40**, a field type **42**, and an associated dimension **44**. The field name **40** is used to identify the information contained in the field. The field type **42** identifies the field as either a dimension, a summary field, or a non-summary field. The dimension field is a search key along which the data is organized and summarized. The summary field is a numeric quantity that provides useful information when summed and averaged. The non-summary field contains information that is associated with each input record, or with a value of the dimension field. The non-summary field would be a field that is not important enough to be a dimension field, or a field that is directly related to an existing dimension field. The associated dimension **44** is used for non-summary fields to identify the dimension field that the information is associated with, or "Detail" if the non-summary information is unique for each input record.

The example in FIG. **2** illustrates a personnel database, where each record **48** in an employee file **50** represents an employee of a company. The input files consist of the employee file **50**, which contains data about each employee, and the department file **52**, which contains information about the different departments. The department file **52** can be joined to the employee file **50** using the common Dept. Id. field. The data dictionary **18** describes the various fields of the input files, and identifies the fields as dimension fields, summary fields and non-summary fields. The Dept Name and Manager fields are non-summary fields associated with the Dept. Id dimension field. The Address and Name fields are non-summary fields associated with each input record. Alternatively, the address and name fields could have been associated with the Employee Id dimension field, since the Employee Id field is unique for each record.

FIG. **3** shows the major components of the database. The database comprises four parts: a summary tree **28**, key value tables **24**, a detail table **30**, and key info tables **26**.

The key value tables **24** provide mappings between integers **54** and the possible values **56** for the different dimension fields. For each dimension field in the input, there is a key value table, with entries for all values of that dimension field that appear in the input. The key value tables **24** allow dimension values to be represented as compact integers in the other parts of the database; the key value table can be indexed to convert these integers **54** into the actual dimension values **56**. In the preferred embodiment of the invention, the key value tables **24** are sorted according to their natural sorting order, so that sorting the numbers associated with a key will result in sorting the key. In this example, the natural sorting order is alphabetical for alphanumeric dimension fields.

The detail table **30** is a relational table representing the input data in a tabular format. For each record in the input, there is an entry **68** in the table, containing dimension fields **58**, summary data fields **60**, and those non-summary data fields that are not associated with dimension fields. The dimension fields **58** are represented numerically as defined by their key value tables **24** to reduce the storage space needed.

The key info tables **26** are used to store information for non-summary fields associated with dimension fields. For each dimension field that has non-summary information associated with it, there exists a key info table **64**. The key info table **64** is indexed **66** using the numeric ordering of the key as defined by the appropriate key value table **70**. For each value of the dimension field, there is an entry in the key info table containing the information associated with that dimension value.

The summary tree **28** is used to summarize and index the detail table **30**.

Referring to FIG. **4**, a root summary node **72** summarizes the set of all records in the detail table **30**, while lower levels **74**, **76** summarize smaller sets, which are defined by the combination of different dimension values **78**. Leaves **80** of the summary tree **72** are pointers to the detail table **30**. The same detail record may appear several times in the summary tree **72**. A node of the tree

represents the set of detail records that are descended from that node. A summary node **82** will summarize the set of records, while a dimension node **84** will index that set of records along a certain dimension field, forming smaller sets. When the set is smaller than a certain threshold, the dimension nodes are replaced by a single detail index node **86**, which contains pointers **80** to individual data records, thus reducing the branching of the summary tree. A detail index node **86** also replaces dimension nodes when the combination of dimension values **78** defining the set in the summary node includes all but one dimension value; the final dimension value would divide the set into very small subsets, which are not worth summarizing in the summary tree. This replacement should be made higher up in the tree if ancestor dimension nodes describe the exact same set as this detail index node; the intermediate dimension and summary nodes do not subdivide the set.

The summary tree **28** of FIG. **1** consists of three types of nodes: summary nodes **82**, dimension nodes **84**, and detail index nodes **86**. Summary nodes **82** contain summary information, while dimension nodes **84** and detail index nodes **86** simply provide structure to the tree.

As shown in FIG. **6**, a summary node contains summary information for the set of records it represents. The summary node contains a count of records in the set **90**, and summary information for each summary field **92**, **94**. This summary information would be the sum **96**, the sum of squares **98**, the minimum **100** and the maximum **102** of the set of values for the summary field.

The summary tree **28**, as shown in FIG. **4**, is structured so that the first level of the summary tree is a summary node **72**, followed by alternating levels of dimension nodes and summary nodes until a detail index node is reached. If a summary node **104** represents fewer than a given threshold number of detail records, or if the summary node represents a set of records that differ in only one dimension field, the child of a summary node is a detail index node **106**. Otherwise, the children of a summary node **108** are dimension nodes **110**, one for each dimension field that has not yet been specified in the set **112** that the summary node represents. These dimension nodes **110** represent the same set as the summary node, causing detail records to be duplicated among these dimension nodes **110**. The children of a dimension node **114** are summary nodes **76**, one for each value of the given dimension in the set represented by the dimension node **114**. These summary nodes **76** represent fewer detail records than the dimension node **114**.

When a summary node **108** has dimension nodes **110** as children, each dimension node represents the same set of detail records. Each detail record in the set is a descendent of all the dimension nodes **110**, creating duplication in the tree. If the tree summarizes n dimensions, the first level of dimension nodes **116** will cause detail records to appear n times. The second level will cause detail records to appear $n(n-1)$ times. If the tree were fully developed, the dimension branching would cause each record to appear in the tree n! times. The detail index nodes reduce this number by cutting off the dimension nodes at a certain level.

FIG. **5** illustrates another technique for eliminating redundancy. Subtrees of the summary tree are shared where possible. In the summary tree, a summary node representing the same set of records appears in several places of the tree, depending on the order of dimensions used to access it. For instance, the summary node **120** representing the set of records with dimension field SEX having value M and dimension field ZIPCODE having value **02046** is in a different part of the tree than the summary node representing records with dimension field ZIPCODE having value **02046** and dimension field SEX having value M. The corresponding subtree **88** could be shared among different parts of the tree **122**, **124**, reducing the duplication.

In the preferred embodiment of the invention, the summary tree is represented using tables. This provides a compact representation of the tree, and provides locality of reference for accessing brother summary nodes under the same dimension node.

FIG. **7** shows the correspondence between the logical structure of the summary tree and a tabular representation of the tree. The root summary node **72** can be represented as a table with a single row. The row contains a count **90** of records in the database, summary information for the summary fields **92**, **94**, and dimension pointers **126** to tables representing the child dimension nodes.

A dimension node **128** and its child summary nodes **136** can be represented using a two-dimensional table. Each row represents a summary node; each row contains the dimension value **138** which identifies the summary node, the count **90** of records in the set, the summary information for the summary fields, and dimension pointers **126** to tables representing child dimension nodes or detail index nodes.

A flow chart for the builder program **20** is shown in FIG. **8**. The builder program makes several passes on the input, generating the different parts of the summary tree database. Step **140** analyzes the input file **16** to determine the types, sizes and value ranges for the input fields. Step **142** creates the key value tables **24** by reading the input, and maintaining tables for each dimension field in main memory **12** of unique dimension values. These tables are sorted, and written out as key value tables **24**. Step **144** creates the key info tables **22**, by reading the input file **16**, and storing the non-summary field values into the appropriate key info table. Step **146** creates the detail table **30** by reading each input record, translating the dimension values into their corresponding numeric indexes using the key value tables **24**, and storing the dimension values, the summary fields, and the appropriate non-summary fields into the detail table **30**.

The detail table **30** is sorted to provide some locality of reference, so that similar entries are located in the same area of secondary storage. In the preferred embodiment of the invention, this is a multiple-key sort on all dimension fields, where the dimension fields are ordered by the number of unique values they have; the dimension field with the least number of unique values is the primary key, while the dimension field with the most number of values is the least significant key. This means that detail records near each other in the detail tables share the most number of possible dimension values. When reading in from secondary storage all records that match a certain combination of dimension values, these related records can be read at one time without performing too many expensive disk seeks.

Step **148** creates the summary tree **28** itself. This is a recursive process that creates the tree in a top-down manner and summarizes the sets in a bottom-up manner. FIG. **9** illustrates this step in more detail. Step **150** selects all records in the detail table **30**. Block **152** represents the recursive process of summarizing a set of re-

cords and producing summary information; it is used recursively at Step **168**. Step **154** checks if the number of records is smaller than the given threshold, or no dimension fields are left to summarize on; if so, Step **178** builds a detail index node, using the record numbers of the current set of records. Step **180** reads in the set of records from the detail table, and calculates summary information from this set.

If the number of records is larger than the threshold and dimension fields remain, for each remaining dimension field **156** the records are sorted along that dimension field **160**, and the records are divided into subsets according to different values of that dimension field **162**. For each subset, the current summary tree built so far is examined in Decision Step **166** to see if the same subset has been built using a different order of dimension fields. If so, a pointer to the appropriate subtree is placed into this part of the tree in Step **176** to accomplish the sharing described in FIG. **5**. If the subset has not yet been processed, Step **168** is a recursive call to the current procedure to summarize this subset, and build a subtree and a summary node in Step **170**. This process continues for all subsets of records in Decision Step **172**. The summary information of these subsets are combined in Step **204** to generate summary information for the current set of records. Sum values are added together, and minimums and maximums combined to create the contents of a summary node. This summary node is returned from the recursive procedure **152** to be placed in the proper place in the summary tree.

FIG. **10** illustrates the logic of the Diver program **32** and how it uses the database **22** to present summary information to the End user **36** given a set of dimension selections **34**. The dimension selections **34** specifies dimension values for a set of dimension fields, selecting a set of detail records. At Step **182**, the root summary node **72** of the summary tree **28** is read into main memory **12**. Step **184** chooses a dimension from the dimension selections. Step **186** follows the appropriate dimension pointer to its target node. If this node is a detail index node **188**, then this detail index node contains a superset of the desired set. Step **200** reads the detail records from the detail table **30**, which are then sorted **202** according to their dimension values of the dimension selections **34**. The subset of records that match the dimension selections **34** are selected and summarized to produce the desired summary information.

If the target node of Step **186** is a not a detail index node, it is a dimension node. Step **190** finds a child summary node of this dimension node that has the dimension value that matches the dimension selection. If no summary node exists **192**, no records match **198** the dimension selections, and the appropriate display is generated **196**. If the summary node exists and more dimension selections remain, Steps **184** through **192** are repeated until all dimension selections have been used. When all dimension selections have been used, the summary node contains the summary information for the desired input set. The appropriate display **196** is generated to present this summary information and other derived information, such as averages and standard deviations, to the user.

While the invention has been shown and described with reference to the preferred embodiment thereof, it will be understood by those skilled in the art that the above and other changes in form and detail may be made therein without departing from the spirit and scope of the invention.

We claim:

1. A data management system for building a database, comprising:

an input record memory for storing a plurality of input records, each input record including a plurality of data fields containing field values;

a database structure memory for storing database structures, the database structures including

a detail table,

a summary tree,

a detail index, and

a summary table;

a record input controller for entering the plurality of input records into the input record memory;

a processor connected from the record input controller and to the input record memory and to the database structure memory for performing operations on the input records and on the database structures; and

a builder control connected to the processor for controlling operations of the processor for building the database structures, including

an input record analyzer control connected from the input record memory for directing the processor for reading and analyzing the data fields of the input records;

a detail table control for directing the processor for constructing the detail table, including

generating a database record corresponding to each input record, assigning a record pointer for each of the database records, each database record being addressable by the assigned record pointer and including dimension fields containing dimension values and summary fields containing numeric information, and

writing the database records into the detail table,

a summary tree control for directing the processor for constructing the summary table, including

reading the database records and selecting summary sets of the plurality of database records wherein each summary set includes a plurality of database records having a common combination of dimension values for the associated dimension fields,

generating summary nodes of the summary table, the summary nodes for storing summary information of the summary fields of the database records of the summary sets of the database records, and

generating summary information from the numeric information contained in the summary fields of the database records and writing the summary information into the summary nodes,

constructing the detail index, including

reading the database records and selecting index sets of the plurality of database records wherein each index set includes a plurality of database records having a common combination of dimension values for the associated dimension fields, and

storing the record pointers assigned to the database records of the index sets in detail index nodes of the detail index; and,

constructing the summary tree, including

generating a summary tree, and

writing the plurality of summary nodes and the plurality of detail index nodes based on combina-

tions of dimension values into the summary tree and arranged in a hierarchical fashion.

2. The system of claim 1, further comprising:

a dictionary memory for storing a data dictionary containing an entry for each type of field in the input records, each data dictionary entry containing information identifying the corresponding type of field in the input records as a dimension field or a summary field; and wherein the input record analyzer control directs the processor for analyzing each input record field, including

reading each field of each input record,

reading the corresponding entry of the data dictionary, and

determining when an input record field is a dimension field and when an input record field is a summary field.

3. The system of claim 1 further comprising:

a key value table memory for storing a key value table,

the key value table containing an entry corresponding to each dimension field value and each entry containing an integer associated with the corresponding dimension field value, and wherein

the builder control further directs the processor for replacing each dimension value in a dimension field of each input record with the associated integer value in the corresponding dimension field of the corresponding database record, including

reading each dimension value of each dimension field of each input record,

reading the associated integer from the corresponding entry of the key value table, and

writing the associated integer into the corresponding dimension field of the corresponding database record.

4. The system of claim 3 wherein the system further comprises:

a dictionary memory for storing a data dictionary containing an entry for each type of field in the input records,

each data dictionary entry identifying an input record field as a summary field or as a non-summary field and containing information associating each non-summary field with the dimension fields; and

a key info memory for storing a key info table, and wherein

the builder control further includes a key info table control for directing

the operations of the processor for

constructing a key info table, and

writing the non-summary field values into the key info table, wherein each non-summary field value is identified by the integer value of an associated dimension field.

5. The system of claim 1 wherein the summary tree further comprises a search tree for accessing the plurality of summary nodes and the plurality of detail index nodes based on combinations of dimension values, the search tree comprising:

a plurality of dimension nodes for identifying sets of records according to combinations of dimension values,

the summary nodes, and

the detail index nodes, wherein

the dimension nodes, the summary nodes and the detail index nodes are arranged in a hierarchical fashion, and wherein

each index set includes a plurality of sets of detail records defined by first combinations of dimension values,

each summary set includes a plurality of sets of the detail records defined by second combinations of dimension values,

the summary nodes are divided into

a first plurality of summary nodes for storing the summarizing records for sets that are summary sets but not index sets, and

a second plurality of summary nodes for storing the summarizing records for sets that are both summary sets and index sets and

each summary node of the second plurality of summary nodes contains a pointer to a detail index node,

each summary node of the first plurality of summary nodes contains a pointer to a dimension node, there being a dimension node for each dimension field that is not specified in combination of dimension

values for said each summary node and the dimension node being associated with the dimension field, and wherein

each dimension node stores a pointer to a child summary node, there being a child summary node for each dimension value contained in a summary set of records summarized by a parent summary node of the dimension node, each dimension value for dimension field being associated with each dimension node, and the child summary node summarizing the subset of the summary set of records containing each dimension value.

6. The system of claim 5 wherein the summary tree control further comprises:

a dimension field selection control for directing the processor for selecting a dimension field,

the dimension field selection control being responsive to a recursive selection control for selecting successive dimension fields;

a sorting control responsive to the dimension field selection control for directing the processor for sorting the records of the detail table according to the selected dimension field;

a dimension field value selection control responsive to the sorting control for directing the processor for selecting each set of records formed by having a common dimension field value for the selected dimension field;

a size control responsive to the dimension field value selection control for directing the processor for determining when a set of records of the sets of records formed by having a common dimension field value for the selected dimension field contains a number of records exceeding a given threshold value;

a detail index generation control responsive to the size control for directing the processor for generating a detail index node containing record serial numbers of each set of records formed by having a common dimension field value for the selected dimension field and containing a number of records less than the threshold value and calculating the summary information for each of the sets of re-

cords formed by having a common dimension field value for the selected dimension field and containing a number of records less than the threshold value;

the recursive selection control, wherein

the recursive selection control is responsive to the detail index generation control for directing the dimension field selection control for directing the processor for selecting a next dimension field when the size of the set exceeds the threshold;

a summary control responsive to the size control for directing the processor for gathering and combining summary information from each set to generate summary information for records when a set contains a number of records greater than the threshold, and

a summary write control for directing the processor for storing the summary information in a summary node.

7. The system of claim **1** wherein said summary information in each summary node comprises:

statistical information including a count of the number of records associated with a summary set of records, and for each summary field, a sum of the values, a sum of the squares of the values, a minimum value, and a maximum value of the values stored in the summary fields of the plurality of database records.

* * * * *